

- **Prénom : Kenza**
- **Nom : HALIL**
- **Groupe : Euros**
- **Promotion : BUT 1 INFORMATIQUE**
- **Nom de la SAE : SAE 201**

Création d'une base de données : Freedom in the world

- **Sommaire :**

1 . Script manuel de création de la base de données

- a. Script SQL de création des tables

2 . Modélisation et script de création avec « AGL »

- a. Illustration comparatives cours/AGL commentée d'une association maillée.
- b. Illustration comparatives cours/AGL commentée d'une association fonctionnelle.
- c. Modèle physique de données réalisé avec AGL
- d. Script SQL de création des tables généré automatiquement par l'AGL
- e. Discussion sur les différences entre les scripts produits manuellement et automatiquement

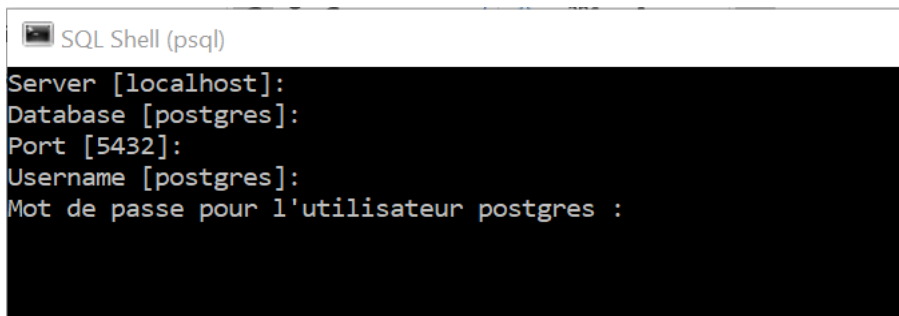
3 . Peuplement des tables

- a. Description commentée des différentes étapes de mon script de peuplement

En premier j'ai installé les logiciels que j'ai besoin dans cette SAE :

- **MySQL Workbench 8.0 CE**
- **PostgreSQL version 16.**

J'ai utilisé « SQL Shell » comme terminal pour exécuter les requêtes



```
SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Mot de passe pour l'utilisateur postgres :
```

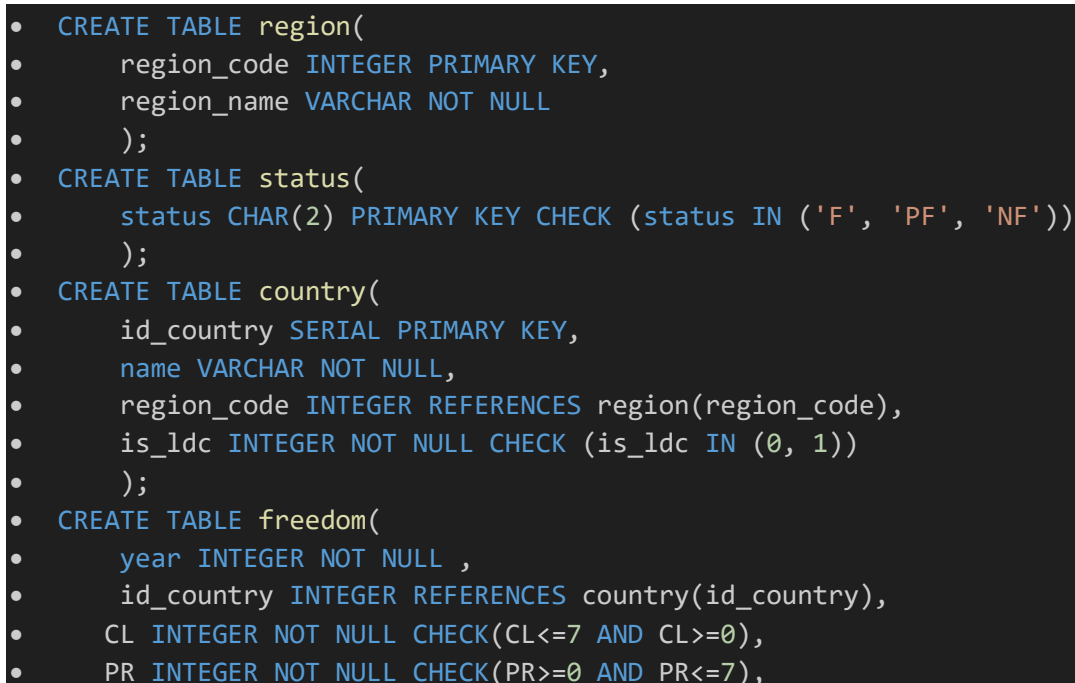
Pour pouvoir se connecter il faut cliquer sur la clé (entrer) 4 fois après il va demander un mot de passe, il faut entrer le mot de passe que nous avons fourni lors de l'installation de PostgreSQL.

1.Script manuel de création de la base de données :

a. Script SQL de création des tables :

Tout d'abord j'ai créé un dossier sur le bureau, je l'ai nommé « SAE Bases de données » puis j'ai créé un fichier « BD » avec l'extension « .sql » avec « Visual studio code » j'ai commencé à créer les tables « FREEDOM, COUNTRY, REGION , STATUS » de la manière suivante :

- **Le script : (copier-coller)**



```
• CREATE TABLE region(
•     region_code INTEGER PRIMARY KEY,
•     region_name VARCHAR NOT NULL
• );
• CREATE TABLE status(
•     status CHAR(2) PRIMARY KEY CHECK (status IN ('F', 'PF', 'NF'))
• );
• CREATE TABLE country(
•     id_country SERIAL PRIMARY KEY,
•     name VARCHAR NOT NULL,
•     region_code INTEGER REFERENCES region(region_code),
•     is_ldc INTEGER NOT NULL CHECK (is_ldc IN (0, 1))
• );
• CREATE TABLE freedom(
•     year INTEGER NOT NULL ,
•     id_country INTEGER REFERENCES country(id_country),
•     CL INTEGER NOT NULL CHECK(CL<=7 AND CL>=0),
•     PR INTEGER NOT NULL CHECK(PR>=0 AND PR<=7),
```

```

• status CHAR(2) REFERENCES status (status),
• PRIMARY KEY(id_country,year)
• );

```

Maintenant je vais expliquer en détails les étapes de création de chaque table :

- La table « REGION » :

```

• CREATE TABLE region(
•     region_code INTEGER PRIMARY KEY,
•     Region_name VARCHAR NOT NULL
• );

```

CREATE TABLE region : indique que on crée une nouvelle table nommée « region »

region_code INTEGER PRIMARY KEY : c'est la première colonne de la table region qui est nommée « region_code » et qui représente une clé primaire « PRIMARY KEY » qui s'agit d'un entier « INTEGER »

region_name VARCHAR NOT NULL : cette syntaxe définit la deuxième colonne de la table region .Il s'agit d'une chaîne de caractère limitée (2<haîne de caractères<2000) et elle ne peut pas contenir de valeurs nulles (NON NULL).

- La table « STATUS »

```

• CREATE TABLE status(
•     status CHAR(2) PRIMARY KEY
• );

```

CREATE TABLE status : indique que on crée une nouvelle table nommée « status »

Status : c'est une chaîne de caractère limitée à 2 caractères (CHAR (2)) et elle a que 3 possibilités {F, PF, NF}

- La table « COUNTRY » :

```

• CREATE TABLE country(
•     id_country SERIAL PRIMARY KEY,
•     name VARCHAR NOT NULL,
•     region_code INTEGER REFERENCES region(region_code),
•     is_ldc INTEGER NOT NULL CHECK (is_ldc IN (0, 1))
• );

```

CREATE TABLE country : indique que on crée une nouvelle table nommée « country »

id_country : c'est la clé primaire « PRIMARY KEY » et « SERIAL » aide à la créer automatiquement en incrémentant de 1 à chaque fois qu'on crée une ligne de la table (auto-incrémentation).

name : c'est du type (VARCHAR) qui ne doit pas être nul.

region_code : c'est une clé étrangère qui fait référence à une clé primaire (region_code) dans la table region à l'aide de l'instruction « REFERENCES »

is_ldc : c'est un entier non nul. Sois 1 sois 0.

- La table « Freedom » :

-

```

• CREATE TABLE freedom(

```

```

• year INTEGER ,
• id_country INTEGER REFERENCES country(id_country),
• CL INTEGER NOT NULL CHECK(CL<=7 AND CL>=0),
• PR INTEGER NOT NULL CHECK(PR>=0 AND PR<=7),
• status CHAR(2) REFERENCES status (status),
• PRIMARY KEY(id_country,year)
• );

```

CREATE TABLE Freedom : indique que on crée une nouvelle table nommée « Freedom »
Year : C'est un entier (INTEGER)

Id_country : c'est un entier qui fait référence à id_country de la table country.

Ici year et id_country sont les deux des clés primaires de la table freedom mais on n'a pas le droit d'utiliser PRIMARY KEY deux fois c'est pour ça on doit utiliser une contrainte de table.

Ce qui est définit c'est :

PRIMARY KEY (id_country, year) : ici on a assemblé les deux clés primaires, avec une contrainte de table.

cl et pl: on a utilisé « CHECK » pour mettre une condition qui est : l'entier doit être inclus dans l'intervalle [0,7]

Et pour le reste c'est le même principe.

On remarque que chaque colonne est indépendante de l'autre et qu'elles sont séparées par une virgule et chaque fois qu'on finisse la création d'une table on met un point-virgule.

Ensuite, j'ai testé mon script s'il est bien fait en utilisant un « SQL shell » : j'ai copié collé mon script dans le terminal de SQL puis je l'ai exécuté, il a renvoyé un message indiquant que la table a bien été créée.

J'ai créé une autre table temporaire je l'ai appelé tompo en utilisant cette commande

```

postgres=# CREATE TABLE tompo(country VARCHAR(277) NOT NULL, year VARCHAR(100) NOT NULL, CL INTEGER NOT NULL CHECK(CL<=7 AND CL>=0), PR INTEGER NOT NULL CHECK(PR>=0 AND CL<=7),status CHAR(2) PRIMARY KEY CHECK (status IN ('F', 'PF', 'NF')), region_code INTEGER NOT NULL, region_name VARCHAR(100) NOT NULL, is_ldc INTEGER NOT NULL CHECK (is_ldc IN (0, 1)));
CREATE TABLE

```

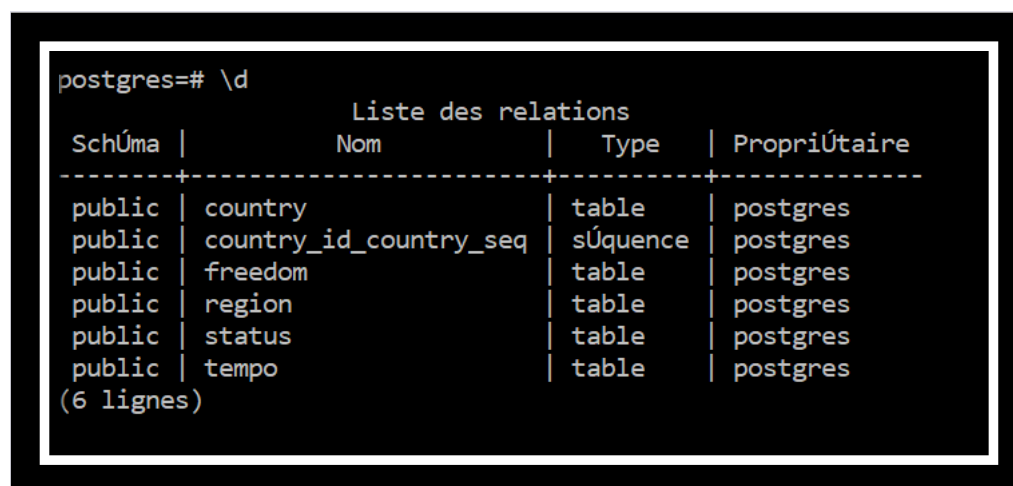
Puis j'ai copié les données du grand fichier dans cette table en utilisant cette commande :

- `\COPY tempo(country, year, CL, PR, status, region_code, region_name, is_idc) FROM '\Users\User\Desktop\freedom.csv' WITH CSV HEADER;`

Remarque(j'ai enregistré le fichier que j'ai téléchargé sur le bureau sous le nom freedom.csv.)

J'ai utilisé la commande : `\d` dans le terminal « **SQL SHELL** » pour afficher les tables que j'ai créé.

Le résultat :



Liste des relations			
Schéma	Nom	Type	Propriétaire
public	country	table	postgres
public	country_id_country_seq	séquence	postgres
public	freedom	table	postgres
public	region	table	postgres
public	status	table	postgres
public	tempo	table	postgres

(6 lignes)

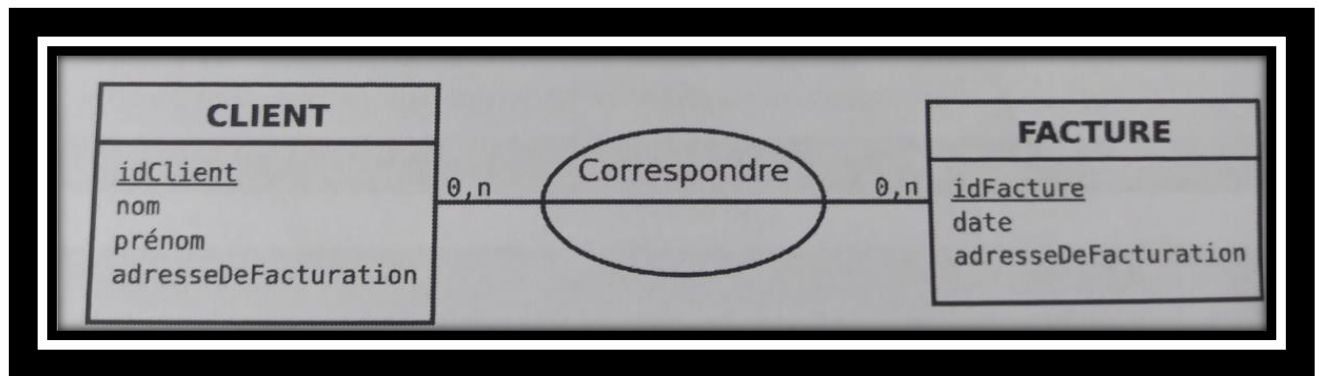
La ligne : **public | country_id_country_seq | séquence | postgres**

Indique qu'il existe une séquence appelée "country_id_country_seq" dans le schéma "public" de la base de données PostgreSQL, et cette séquence est associée à une colonne auto-incrémentée utilisée pour générer des identifiants uniques dans « country ».

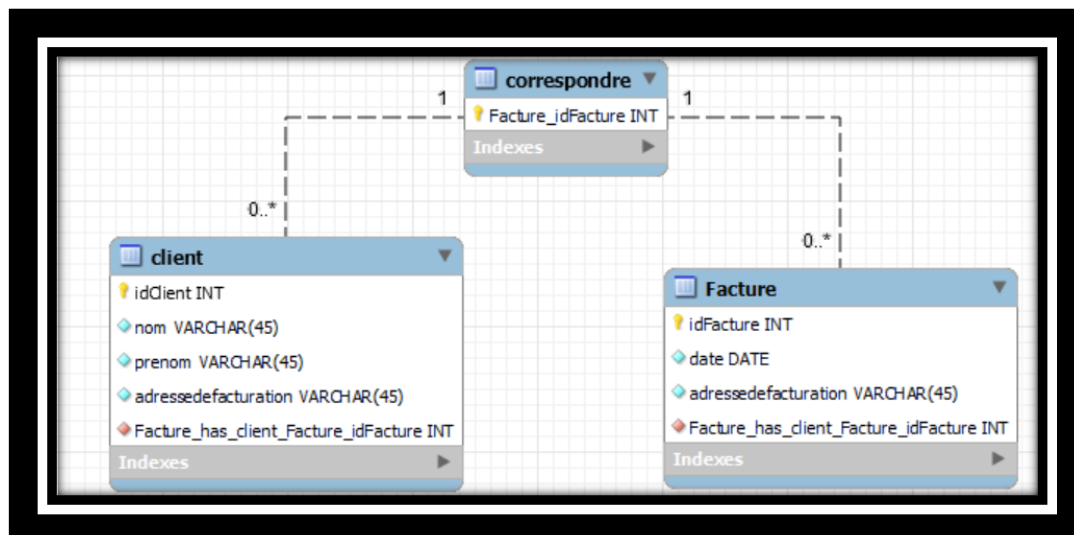
2. Modélisation et script de création avec « AGL » :

a. Illustration comparatives cours/AGL commentée d'une association maillée :

- MCD association maillée :

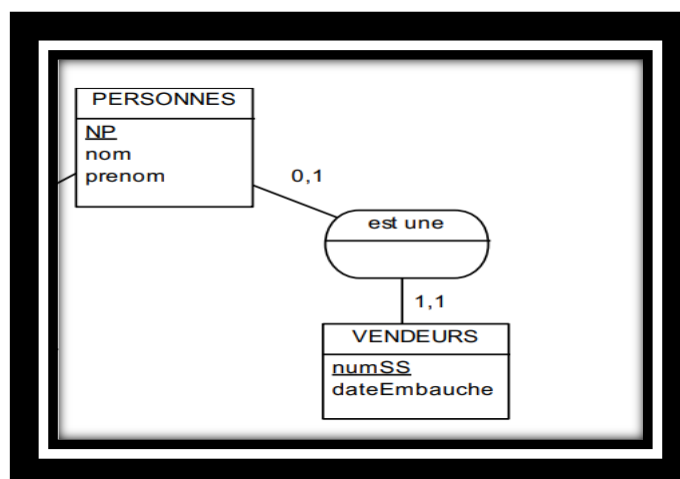


- AGL association maillée :

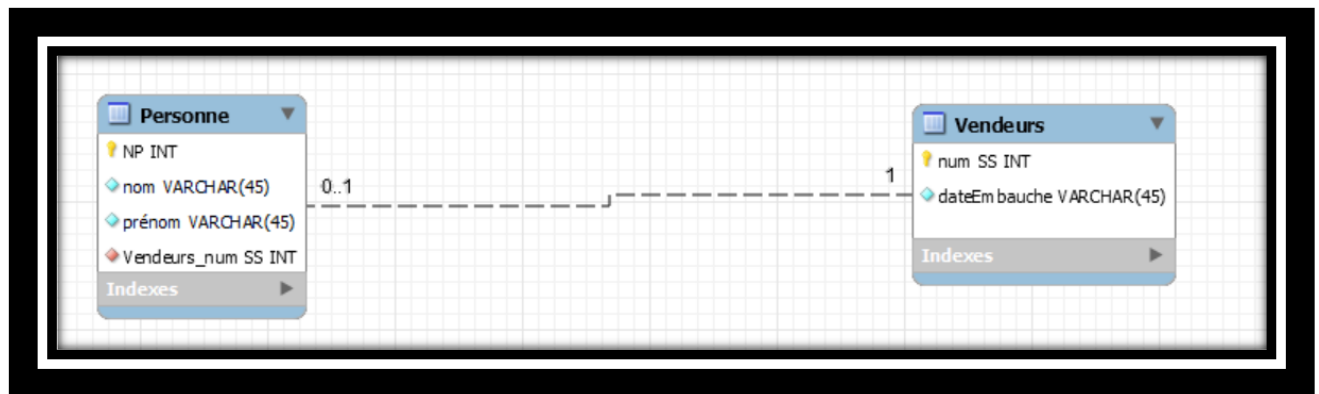


b. Illustration comparatives cours/AGL commentée d'une association fonctionnelle.

- MCD association fonctionnelle :



- AGL association fonctionnelle :



Les points distinctifs pour chacun de : « MCD » et « AGL »

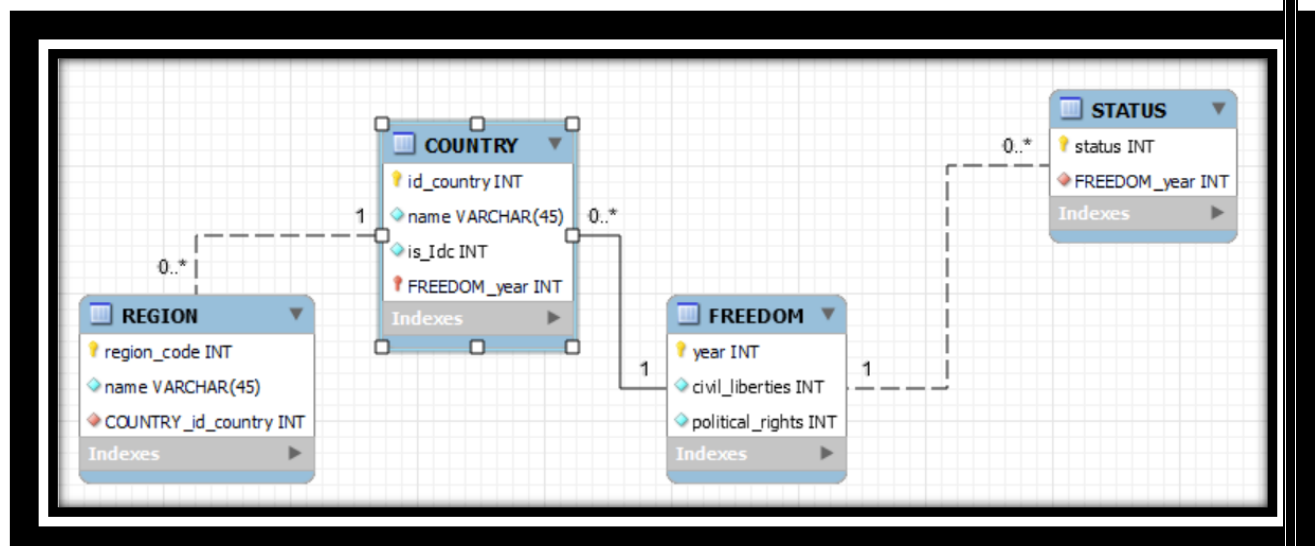
✓ Le « MCD » :

- Avec MCD il affiche que le nom des attributs.
- Un MCD pourrait être utilisé pour représenter les entités ainsi que les relations entre elles.
- le MCD est spécifiquement orienté vers la modélisation de données.
- Avec le MCD un attribut souligné représente une clé primaire.

✓ L' « AGL » :

- Avec l'AGL il s'affiche même le type de l'attribut
- L'AGL est un ensemble d'outils intégrés utilisés pour divers aspects du développement logiciel
- Avec AGL il y'a des signes qui représentent le type de l'attribut (clé primaire, clé étrangère et primaire au même temps ...etc.)
- Avec l'AGL la ligne continue représente une relation permanente entre deux entités et une ligne discontinue représente une relation plus faible ou optionnelle qui peut être conditionnelle ou temporaire.

c. Modèle physique de données réalisé avec AGL



d. Script SQL de création des tables généré automatiquement par l'AGL :

Script généré automatiquement :

```
1  -- MySQL Script generated by MySQL Workbench
2  -- Wed Jan 3 13:34:48 2024
3  -- Model: New Model   Version: 1.0
4  -- MySQL Workbench Forward Engineering
5
6  SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
7  SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
8  SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_
9
10 -----
11 -- Schema mydb
12 -----
13
14 -----
15 -- Schema mydb
16 -----
17 CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
18 -----
19 -- Schema new_schema1
```

```
23 -----
24 -- Table `mydb`.`FREEDOM`
25 -----
26 CREATE TABLE IF NOT EXISTS `mydb`.`FREEDOM` (
27   `year` INT ZEROFILL NOT NULL,
28   `civil_liberties` INT NOT NULL,
29   `political_rights` INT NOT NULL,
30   PRIMARY KEY (`year`))
31 ENGINE = InnoDB;
32
33 -----
34 -- Table `mydb`.`COUNTRY`
35 -----
36
37 CREATE TABLE IF NOT EXISTS `mydb`.`COUNTRY` (
38   `id_country` INT NOT NULL,
39   `name` VARCHAR(45) NOT NULL,
40   `is_idc` INT NOT NULL,
```

```
41   `FREEDOM_year` INT ZEROFILL NOT NULL,
42   PRIMARY KEY (`id_country`, `FREEDOM_year`),
43   INDEX `fk_COUNTRY_FREEDOM1_idx` (`FREEDOM_year` ASC) VISIBLE,
44   CONSTRAINT `fk_COUNTRY_FREEDOM1`
45     FOREIGN KEY (`FREEDOM_year`)
46     REFERENCES `mydb`.`FREEDOM` (`year`)
47     ON DELETE NO ACTION
48     ON UPDATE NO ACTION)
49 ENGINE = InnoDB;
50
51 -----
52 -- Table `mydb`.`REGION`
53 -----
54
55 CREATE TABLE IF NOT EXISTS `mydb`.`REGION` (
56   `region_code` INT NOT NULL,
57   `name` VARCHAR(45) NOT NULL,
58   `COUNTRY_id_country` INT NOT NULL,
59   PRIMARY KEY (`region_code`),
```


e. Discussion sur les différences entre les scripts produits manuellement et automatiquement :

- Le script automatique généré par l'AGL est trop long et il contient trop de détails par rapport au script manuel.
- Le script généré par l'AGL est produit à partir d'un schéma visuel créé dans l'outil mais le script manuel est écrit par un développeur manuellement.

```
1  -- MySQL Script generated by MySQL Workbench
2  -- Wed Jan 3 13:34:48 2024
3  -- Model: New Model   Version: 1.0
4  -- MySQL Workbench Forward Engineering
```

- Ces 4 premières lignes sont présentes dans un script généré par l'AGL elles représentent :

• -- MySQL Script generated by MySQL Workbench	○ Indique que le script SQL a été généré par MySQL Workbench.
• -- Wed Jan 3 13:34:48 2024	○ Donne la date et l'heure précises à laquelle le script a été généré.
• -- Model: New Model Version: 1.0	○ Indique le nom du modèle (dans cet exemple, "New Model") et sa version (1.0)
• -- MySQL Workbench Forward Engineering	○ Signale que le script contient des instructions pour effectuer un "Forward Engineering" avec MySQL Workbench.

Et on ne trouve pas des lignes pareilles dans un script manuel.

```
6  SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
7  SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
8  SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_
```

- Ces trois lignes sont des lignes qu'on trouve dans un script automatique, elles représentent des instructions SQL permettant de modifier temporairement des paramètres de configurations dans MYSQL, mais on ne les trouve pas dans un script manuel.

```
7  CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
```

- Cette instruction SQL crée une base de données nommée **mydb** avec la clause **IF NOT EXISTS** et spécifie le jeu de caractères par défaut comme étant UTF-8, et on ne trouve pas cette ligne dans le script manuel.

```

23 -----
24 -- Table `mydb`.`FREEDOM`
25 -----
26 CREATE TABLE IF NOT EXISTS `mydb`.`FREEDOM` (
27   `year` INT ZEROFILL NOT NULL,
28   `civil_liberties` INT NOT NULL,
29   `political_rights` INT NOT NULL,
30   PRIMARY KEY (`year`))
31 ENGINE = InnoDB;

```

```

• CREATE TABLE freedom(
•   Year INTEGER NOT NULL,
•   id_country INTEGER REFERENCES country(id_country),
•   CL INTEGER NOT NULL CHECK(CL<=7 AND CL>=0),
•   political_rights INTEGER NOT NULL CHECK(PR>=0 AND PR<=7),
•   status CHAR(2) REFERENCES status (status),
•   PRIMARY KEY(id_country,year)
• );

```

- On remarque qu'avec AGL ça commence avec « Table 'mydb'.FREEDOM' » alors que dans le script manuel on a directement commencé par créer la table avec la commande « CREATE TABLE » puis dans le script automatique on remarque que il est écrit CREATE TABLE IF NOT EXISTS automatiquement mais manuellement si on exécute notre script une erreur s'affiche si une table existe déjà sauf si on ajoute IF NOT EXISTS.
- On voit aussi que dans le script manuel on a défini la clé primaire avec (INTEGER PRIMARY KEY) mais le script automatique au lieu de INTEGER s'est écrit INT seulement et « ZEROFILL », cette option spécifie que les zéros seront ajoutés à gauche des nombres entiers pour remplir la largeur totale spécifiée. Par exemple, si la largeur totale est de 4 chiffres et la valeur est 12, la valeur stockée sera "0012".
- Les noms des colonnes dans un script automatique sont mis entre (') sauf que ce n'est pas la même chose avec un script manuel.
- On remarque que les virgules qui séparent les colonnes sont communes entre les deux scripts.

3. Peuplement des tables

a. Description commentée des différentes étapes de mon script de peuplement

Un script de peuplement sert à remplir les tables créées par des données stockées dans le fichier donné dans la SAE, ou chaque table est indépendante des autres et contient que les données reliées aux attributs de chaque table.

J'ai créé une autre table temporaire je l'ai appelé tempo en utilisant cette commande :

- **CREATE TABLE tempo(country VARCHAR NOT NULL, year INTEGER NOT NULL, CL INTEGER NOT NULL CHECK(CL<=7 AND CL>=0), PR INTEGER NOT NULL CHECK(PR>=0 AND PR<=7), status VARCHAR NOT NULL, Region_code INTEGER NOT NULL, Region_name VARCHAR NOT NULL, is_idc INTEGER NOT NULL CHECK (is_idc IN (0, 1)));**

puis j'ai copié les données du grand fichier dans cette table en utilisant cette commande :

- **\COPY tempo(country, year, CL, PR, status, region_code, region_name, is_idc) FROM '\Users\User\Desktop\freedom.csv' WITH CSV HEADER;**

Puis on utilise ses commandes :

- **INSERT INTO region (region_code, region_name) SELECT DISTINCT Region_Code, Region_Name FROM tempo ;**
- **INSERT INTO status (status) SELECT DISTINCT Status FROM tempo ;**

Ces commandes sélectionnent des données spécifiques dans la grande table (tempo) et les insérer dans les tables country, region, freedom, status

Exemple :

La table region a comme colonnes : region_code, region_name

Donc on sélectionne (projection) de region_code et region_name puis j'ai utilisé DISTINCT pour éliminer les doublons.

On obtient les tables status et region avec la commande :

- **SELECT * FROM status ;**
- **SELECT * FROM region ;**

```

postgres=# SELECT * FROM status;
status
-----
PF
NF
F
(3 lignes)

postgres=# SELECT * FROM region;
 region_code | region_name
-----+-----
          142 | Asia
           9 | Oceania
          19 | Americas
         150 | Europe
           2 | Africa
(5 lignes)

```

Pour la table country j'ai utilisé cette commande :

- **INSERT INTO country(name, region_code, is_ldc) SELECT DISTINCT country, region_code, is_ldc FROM tompo**

```

postgres=# INSERT INTO country (name, region_code, is_ldc) SELECT DISTINCT country, region_code, is_ldc FROM tompo;
INSERT 0 193
postgres=# SELECT * FROM country;
 id_country | name | region_code | is_ldc
-----+-----+-----+-----
          1 | Monaco | 150 | 0
          2 | Croatia | 150 | 0
          3 | Nauru | 9 | 0
          4 | Zambia | 2 | 1
          5 | Montenegro | 150 | 0
          6 | Indonesia | 142 | 0
          7 | Yemen | 142 | 1
          8 | Romania | 150 | 0
          9 | Iraq | 142 | 0
         10 | Philippines | 142 | 0
         11 | Paraguay | 19 | 0
         12 | Peru | 19 | 0
         13 | Chad | 2 | 1
         14 | Dominican Republic | 19 | 0
         15 | Australia | 9 | 0
         16 | Bahamas | 19 | 0
         17 | Andorra | 150 | 0
         18 | Georgia | 142 | 0
         19 | Kyrgyzstan | 142 | 0
         20 | Austria | 150 | 0
         21 | Benin | 2 | 1
         22 | Djibouti | 2 | 1
         23 | Libya | 2 | 0
         24 | Sri Lanka | 142 | 0
         25 | Oman | 142 | 0
         26 | Bahrain | 142 | 0
         27 | Kuwait | 142 | 0
         28 | Angola | 2 | 1
         29 | Pakistan | 142 | 0
         30 | Mozambique | 2 | 1
         31 | Cambodia | 142 | 1
         32 | Hungary | 150 | 0

```

On remarque que la colonne id_country s'est créé automatiquement et s'auto incrémente avec SERIAL.

Pour le peuplement de la table freedom :

- `INSERT INTO freedom (year, id_country, CL, PR, status) SELECT tempo.year, country.id_country, tempo.CL, tempo.PR, tempo.status FROM tempo JOIN country ON country.name = tempo.country;`

```
postgres=# SELECT * FROM freedom;
 year | id_country | cl | pr | status
-----+-----+---+---+-----
1995  |          62 | 7  | 7  | NF
1996  |          62 | 7  | 7  | NF
1997  |          62 | 7  | 7  | NF
1998  |          62 | 7  | 7  | NF
1999  |          62 | 7  | 7  | NF
2000  |          62 | 7  | 7  | NF
2001  |          62 | 7  | 7  | NF
2002  |          62 | 6  | 6  | NF
2003  |          62 | 6  | 6  | NF
2004  |          62 | 6  | 5  | NF
2005  |          62 | 5  | 5  | PF
2006  |          62 | 5  | 5  | PF
2007  |          62 | 5  | 5  | PF
2008  |          62 | 6  | 5  | NF
2009  |          62 | 6  | 6  | NF
2010  |          62 | 6  | 6  | NF
2011  |          62 | 6  | 6  | NF
2012  |          62 | 6  | 6  | NF
2013  |          62 | 6  | 6  | NF
2014  |          62 | 6  | 6  | NF
2015  |          62 | 6  | 6  | NF
2016  |          62 | 6  | 6  | NF
2017  |          62 | 6  | 5  | NF
2018  |          62 | 6  | 5  | NF
2019  |          62 | 6  | 5  | NF
2020  |          62 | 6  | 5  | NF
1995  |          51 | 4  | 3  | PF
1996  |          51 | 4  | 4  | PF
1997  |          51 | 4  | 4  | PF
1998  |          51 | 5  | 4  | PF
1999  |          51 | 5  | 4  | PF
2000  |          51 | 5  | 4  | PF
2001  |          51 | 4  | 3  | PF
```

la commande copie les données pertinentes de la table `tempo` vers la table `freedom`, en associant les pays à leurs identifiants respectifs de la table `country`. Cela permet de lier les informations de liberté (year, CL, PR, status) avec les pays correspondants à partir de la table `country`.